{快速启动开发参考文档}

REVISION HISTORY

Revision No.	Description	Date
{1. 0. 0}	• {Initial release}	{07/12/2019}
{1. 0. 1}	• {add Lib Usage}	{07/15/2019}

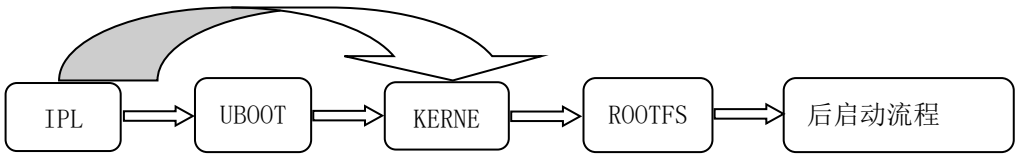
TABLE OF CONTENTS

1. 快速启动的配置

1.1. 概述

本文档将介绍快速启动的配置以及使用。

1.2. 启动流程



1. 默认情况，快速启动将跳过 uboot，直接启动 kernel
2. 需要进入 uboot 可以长按 enter，重新开机直到进入 uboot
3. 后启动流程是指系统起来之后 profile 执行阶段

1.3. 编译

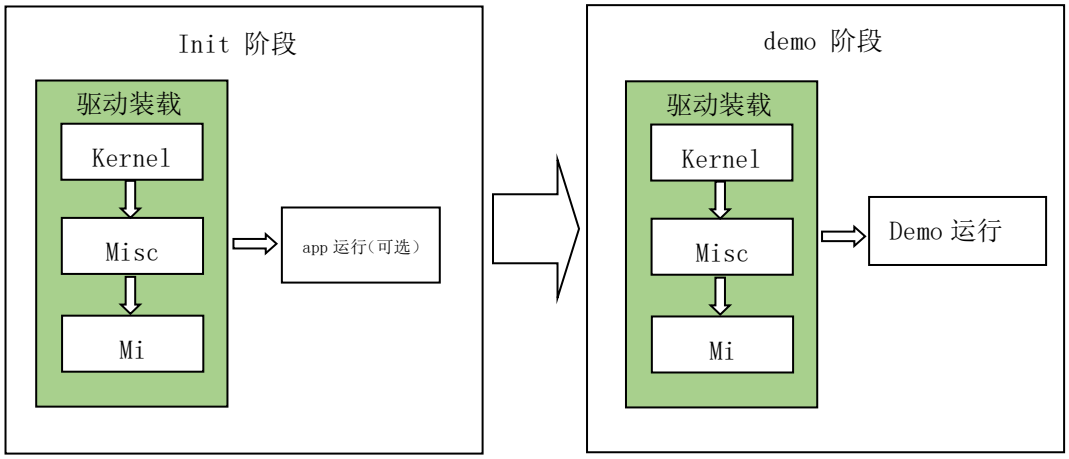
1. 找到相应 ramfs 配置，例：

./setup_config.sh configs/nvr/i2m/8.2.1/nor.glibc-ramfs.011a.64

2. make image

1.4. 后启动流程介绍

这个阶段包括分区的挂载与 modules 的装载，大致分为两个阶段：init 阶段 与 demo 阶段



init 阶段：（refer to /etc/init.sh）

该阶段进行系统必要的环境配置：

1. 装载系统环境配置必要驱动
   依次装载 kernel，misc， mi 驱动

2. app 运行
   该 demo 需在后台运行，存放于 dram

demo 阶段：（refer to demo.sh）

该阶段进行 demo 运行的其它操作

1. 装载次要驱动
   依次装载 kernel，misc， mi 驱动

2. demo 运行
   该 demo 存放于 flash

1.5. 模块装载配置

本小节讲解模块的装载相应的配置文件，可以根据需求灵活配置。

init 阶段：

1. 特点：
   模块存放于 ramfs 分区，系统运行时存放于 dram 中（lib/modules/4.9.84）

2. 配置文件存放目录：
   project 下相应驱动存放目录

3. 配置文件：
   用于配置各阶段装载的驱动，将驱动名添加到相应文件即可
   a. **kernel_mod_list：**
      系统初始化的 kernel 驱动，例如 nls_utf8.ko
   b. **misc_mod_list：**
      系统初始化的 misc 驱动，例如 mhal.ko
   c. **mi_mod_list：**
      系统初始化的 mi 驱动，例如 mi_sys.ko
4. 配置文件使用方式：
   将相应驱动的驱动名添加到对应配置文件

demo 阶段：

1. 特点：
   模块存放于 config 分区（flash），不占用内存空间（/config/modules/4.9.84）

2. 配置文件存放目录：
   project 下相应驱动存放目录

3. 配置文件：
   用于配置各阶段装载的驱动，将驱动名添加到相应文件即可
   a. **kernel_mod_list_late**：
      demo 运行需要的 kernel 驱动， 例如：usb-storage.ko
   b. **misc_mod_list_late**：
      demo 运行需要的 misc 驱动，例如：fbdev.ko
   c. **mi_mod_list_late**：
      demo 运行需要的 mi 驱动，例如：mi_ai.ko
4. 配置文件使用方式：
   将相应驱动的驱动名添加到对应配置文件

1.6. 动态链接库的存放分区配置

动态链接库可以存放到 ramfs 分区与 config 分区，前者占用内存空间，后者存放于 flash。

存放于 ramfs 分区

1. 配置文件名：lib_list_first
2. 配置文件路径：
   动态链接库目录，例如：release/usbcam/i6/009A-fastboot/glibc/8.2.1/lib/dynamic
3. 使用方式：
   将对应库名添加到配置文件

存放于 config 分区

1. 配置文件名：lib_list_second
2. 配置文件路径：
   动态链接库目录：例如：release/usbcam/i6/009A-fastboot/glibc/8.2.1/lib/dynamic
3. 使用方式
   将对应库名添加到配置文件

1.7. alkaid 与 Image 编译配置

alkaid 编译选项配置：

1. 配置文件位置：image/configs/ 例：configs/usbcam/i6/nor.glibc-ramfs.009a.64.qfn88
2. 配置项：

配置项	作用	实例
CHIP	芯片类型	i6
BOARD	开发板类型	009A-fastboot
BOARD_NAME	开发板名字	SSC009A-S01A
PRODUCT	产品类型	usbcam
TOOLCHAIN	编译工具类型	glibc
TOOLCHAIN_VERSION	编译工具版本	8.2.1
KERNEL_VERSION	linux 内核版本	4.9.84
LIBC	libc 库的版本	libc-2.28

BUSYBOX	指定 busybox	busybox-1.20.2-arm-linux-gnueabihf-glibc-8.2.1-dynamic
IMAGE_CONFIG	image 配置文件	IMAGE_CONFIG = nor.ramfs.hfglibc.nvr.mma
CUSTOMER_OPTIONS	添加 sdk 编译环境	null_options.mk
CUSTOMER_TAILOR	开启/关闭 sdk 编译模块	usbcam_i6_tailor.mk
MMAP	Linux 内存分布配置	MMAP_I6_64.h
MHAL	mhal 编译选项	i6
IQ0- IQ3	所需 Iq 文件配置	imx307_iqfile.bin （ipc）
EXBOOTARGS	拓展 bootargs 选项	loglevel=0
kernel$(BOOTENV)	bootargs kernel 启动配置	LX_MEM=$(KERNEL_MEMLEN) mma_heap=mma_heap_name0,miu=0,sz=0x2000000 mma_memblock_remove=1
TOOLCHAIN_REL	Toolchain 前缀	arm-linux-gnueabihf-
SENSOR_LIST	开发板所需 sensor	imx291_MIPI.ko imx307_MIPI.ko sc4236_MIPI.ko SC4238_MIPI.ko
SENSOR0	默认 sensor	imx307_MIPI.ko
SENSOR0_OPT	sensor 装载参数	chmap=1
FLASH_SIZE	flash 大小	16M

Image 编译配置：

1. 配置文件位置：image/configs/ 例：image/configs/i6/nor.ramfs.hfglibc.nvr.mma
2. 配置项：

配置项	作用	实例
IMAGE_LIST	配置 image 生成的 bin 档目标	ipl uboot kernel rootfs nvrservice customer
FLASH_TYPE	flash 的类型	nor
BOOT_TYPE	rootfs 类型	fastboot
PAT_TABLE	flas 接口类型	spi
PHY_TEST	是否测试 phy	no
ipl$(RESOUCE)	ipl 路径	$(PROJ_ROOT)/board/$(CHIP)/boot/ipl/fastboot/IPL.bin
ipl_cust$(RESOUCE)	ipl_cust 路径	$(PROJ_ROOT)/board/$(CHIP)/boot/ipl/fastboot/IPL_CUST.bin
uboot$(RESOUCE)	uboot 路径	$(PROJ_ROOT)/board/$(CHIP)/boot/$(FLASH_TYPE)/uboot/u-boot.xz.img.bin
kernel$(RESOUCE)	kernel Image 路径	$(PROJ_ROOT)/release/$(PRODUCT)/$(CHIP)/$(BOARD)/$(TOOLCHAIN)/$(TOOLCHAIN_VERSION)/bin/kernel/$(FLASH_TYPE)/uImage.mz
kernel$(PATSIZE)	Kernel 分区大	0x200000

	小	
kernel$(BOOTENV)	bootargs 的 kernel 参数指定	$(KERNEL_BOOT_ENV) loglevel=0
kernel$(BOOTCMD)	bootcmd 的 kernel 参数指定	sf read $(KERNELBOOTADDR) \$${sf_kernel_start} \$${sf_kernel_size}\;
rootfs$(RESOUCE)	rootfs 输出目录	$(OUTPUTDIR)/rootfs
rootfs$(FSTYPE)	rootfs 所用文件系统类型	ramfs
rootfs$(PATSIZE)	rootfs 所占分区大小	0x400000
rootfs$(BOOTENV)	bootargs 的 rootfs 参数指定	rootfstype=ramfs initrd=$(INITRAMFSLOADADDR),$(rootfs$(PATSIZE))
rootfs$(BOOTCMD)	bootcmd 的 rootfs 参数指定	mxp r.info rootfs\; sf read $(INITRAMFSLOADADDR) \$${sf_part_start} \$${sf_part_size}\;

3. 用户分区相关配置

配置项	作用	实例
USR_MOUNT_BLOCKS	配置所需用户分区（支持 config customer）	nvrservice customer
nvrservice$(RESOUCE)	nvrservice 分区目录	$(OUTPUTDIR)/tvconfig/config
nvrservice$(FSTYPE)	nvrservice 分区类型	squashfs
nvrservice$(PATSIZE)	nvrservice 分区大小	0x300000
nvrservice$(MOUNTTG)	nvrservice 分区挂载目标	/config
nvrservice$(MOUNTPT)	nvrservice 分区挂载点	/dev/mtdblock3
customer$(RESOUCE)	customer 分区目录	$(OUTPUTDIR)/customer
customer$(FSTYPE)	customer 分区类型	jffs2
customer$(PATSIZE)	customer 分区大小	0x6B0000
customer$(MOUNTTG)	customer 分区挂载目标	/customer
customer$(MOUNTPT)	customer 分区挂载点	mtd:customer

2. 驱动与库的使用

2.1. 简介

本章将根据所需功能介绍所需要的驱动和一些动态链接库的使用。

2.2. 按驱动分类

所属分类	驱动	备注
usb 基础驱动	usb-comon.ko	usb 基础驱动
	usb-core.ko	usb host 基础驱动
	ehci-hcd.ko	usb2.0 host 控制器驱动
	xhci-hcd.ko	usb3.0 host 控制器驱动
u 盘	usb-storage.ko	usb 盘驱动 （依赖 usb 基础驱动）
鼠标	mousedev.ko	鼠标输入驱动
	usbhid.ko	usb 鼠标（依赖 usb 基础驱动）
网卡	of_mdio.ko	依赖文件
	kdrv_emac.ko	平台相关底层网卡驱动
	sstar_100_phy.ko	
	sunrpc.ko	用于远程命令执行的远程过程调用（RPC）协议，被网络文件系统（NFS）使用
	libphy.ko	网络基础驱动
	fixed_phy.ko	
mtd 设备	ubi.ko	ubi 驱动
mmc card	mmc_core.ko	mmc 基础驱动
	mmc_block.ko	
	kdrv_sdmmc.ko	sd 卡驱动
看门狗	mdrv_wdt.ko	看门狗驱动
字体	nls_utf8.ko	utf8 字体驱动
硬盘	sd_mod.ko	scsi 接口磁盘支持
	libahci.ko	串行 ata 接口相关驱动
	ahci_platform.ko	
	libahci_platform.ko	

	libata.ko	
	mdrv-sata-host.ko	sata 底层驱动
iic	mii.ko	i2c 驱动
文件系统	grace.ko, nfs.ko	nfs 网络文件系统
	grace.ko, nfsv2.ko	nfs 网络文件系统（version2）
	ubifs.ko	ubi 文件系统（flash）
	jffs2.ko	jffs2 文件系统（flash）
	squashfs	压缩只读文件系统
	fat.ko	fat 文件系统
	vfat.ko	vfat 文件系统（u 盘）
	cifs.ko	cifs 文件系统（samba）
	ntfs.ko	ntfs 文件系统

2.3. 按库分类

toolchain 相关	librt-2.28.so	实时扩展库，包含信号量，同步 io 等
	libm-2.28.so	数学相关库
	ld-2.28.so	链接库，
	libresolv-2.28.so	提供网络域名包解析相关功能
	libdl-2.28.so	加载动态
	libstdc++.so	标准库
	libc-2.28.so	
	libpcprofile.so	
	libpthread-2.28.so	线程相关库
MI 相关	libmi_common.so	mi 必备基础库
	libmi_sys.so	mi 必备基础库，负责搭建基础 mi 框架，提供 buf 管理，
	libmi_sensor.so	sensor 相关库
	libmi_vif.so	vif 相关库
	libmi_vpe.so	vpe 相关库
	libmi_venc.so	提供视频解码功能
	libmi_rgn.so	提供画图功能
	libmi_disp.so	提供视频拼接，视频输出功能，对接 panel，hdmi 等
	libmi_vdec.so	提供视频解码功能
	libmi_vdf.so	提供视频算法功能，包括移动监测，遮挡检测，虚拟围栏等

	libmi_shadow.so	提供 vdf 对接接口
	libmi_iqserver.so	提供网络调试 iq 的功能，对接 isp 模块
	libmi_ldc.so	提供鱼眼镜头回正支持
	libmi_ive.so	提供硬件算子的使用
	libmi_isp.so	提供高级图像调试接口
	libmi_ai.so	audio 输入相关库
	libmi_ao.so	audio 输出相关库
	libmi_gfx.so	提供数据搬移功能
	libmi_hdmi.so	hdmi 相关库
	libmi_panel.so	panel 相关库
	libmi_fb.so	提供 fb 设备支持
	libmi_cipher.so	提供数据加密解密功能
others	libcus3a.so	提供 isp 高级功能，如自动对焦
	libMD_LINUX.so	vdf 相关库，提供移动监测功能
	libOD_LINUX.so	vdf 相关库，提供遮挡检测功能
	libVG_LINUX.so	vdf 相关库，提供虚拟围栏功能
	libAEC_LINUX.so	audio 算法库，提供 aec 功能
	libAED_LINUX.so	audio 算法库，提供 babycry 检测，高分贝检测
	libAPC_LINUX.so	audio 算法库，提供降噪，均衡化，自动增益控制，高通滤波
	libSRC_LINUX.so	audio 算法库，提供重采样功能
	libg711.so	audio 算法库，提供 g711 编解码
	libg726.so	audio 算法库，提供 g726 编解码

注意：toolchain 相关库为必须库，mi 库请根据需要加载，others 需根据相应 mi 模块需要加载。